C++ による代数的構造の表現

梅谷 武

作成：2000-03-22 更新：2005-04-20

C++ の数値計算ライブラリを開発するにあたって、C++ によって代数的構造を表現することについて考察します。代数的構造の表現は抽象クラスとして記述され、組み込み型以外のすべてのクラスは、これらから派生させることになります。
IMS:20000322001; NDC:418; keywords:C++, 数値計算, 代数的構造;

目　　次

1 はじめに

1.1 C++ による数値計算ライブラリの開発構想

Pentium あるいは Pentium 互換 CPU が驚異的な計算性能をもつようになったことから、PC 環境で数値計算を本格的にやってみたいというようなことを２〜３年前から考えるようになりました。いろいろ調べた結果、時間がかかっても自分で考えて、独自にコーディングすることが一番勉強になるというように考えるようになりました。

まず小手調べに

- C++ で多倍長整数クラスを書いてみる
- この多倍長整数クラスを Tcl/Tk に拡張オブジェクトとして組み込んでみる

という実験をやってみたいと思います。

将来的な構想としては

- 計算性能を上げるためにアセンブラを使用する
- 手軽に使えるインターフェースをもたせる (Tcl/Tk の採用を検討中)
- 計算結果を 2 次元あるいは 3 次元のコンピュータグラフィックスで表現できる機能をもたせる (OpenGL の採用を検討中)

というようなことを考えていますが、空き時間を利用したプロジェクトですので、マイペースで気長にやっていくつもりです。

1.2 開発環境

開発効率を優先するために、慣れている

- Windows 98/2000
- Visual C++ 6.0
- Pentium 互換

というプラットホームで当面開発を進めます。

1.3 C++ による代数的構造の表現

この文書では、C++ による代数的構造の表現について考察します。こういうことをしなくとも数値計算ライブラリの開発はできるのですが、C++ で、数をどのように計算機上に表現できるのかをじっくり考えておくことは、より良い設計を行なうために重要ではないかと思います。

1.4 命名法

C++ における開発では、いろいろなクラスライブラリを共存させることが多いため、一般名詞を使って名前を付けると衝突してしまうことがよくあります。名前空間という機構もありますが、これだけではうまく衝突を回避できない場合もありますし、これが実装されていない C++ 処理系もありますので、この開発プロジェクトでは、すべてのクラスに「S0_」という接頭辞を付けることにします。最初の「S」はプロジェクト

コードを表わし、次の「0」は版数を表わします。

2 集合型

2.1 集合型 (set)

集合型は集合を表現するものです。集合を表現するには、その有限部分集合の元をある変数の状態として一意的に表現する何らかの方法を考えて、その変数の状態集合をもとの集合へ埋め込む 1 対 1 の写像を定義します。この写像は全射でなくてもいいことにしておきます。C++ の語法に従い、変数の雛型のことをクラス、状態をもつ変数のことをそのクラスの実体と呼ぶことにします。

集合型クラスの任意の実体はもとの集合の元に対応し、任意の 2 元について等しいか等しくないかを判定することができなければなりません。さらに代入演算子が定義されていなければなりません。

```
class S0_Set
{
public:
  virtual bool operator==(const S0_Set&) = 0; // ==演算子（等しい）
  virtual bool operator!=(const S0_Set&) = 0; // !=演算子（等しくない）
  virtual S0_Set& operator=(const S0_Set&) = 0; // =演算子（代入）
};
```

2.2 順序集合型 (ordered set)

順序集合型は、順序集合、すなわち順序関係が定義されている集合を表現します。全順序であることは仮定していませんが、全順序であること、すなわち任意の 2 元が比較可能であることは、C++ ではうまく表現できないので全順序集合も順序集合型として表現することになります。

```
class S0_OrderedSet : public S0_Set
{
public:
  virtual bool operator<(const S0_OrderedSet&) = 0; // <演算子（小さい）
  virtual bool operator>(const S0_OrderedSet&) = 0; // >演算子（大きい）
  virtual bool operator<=(const S0_OrderedSet&) = 0;// <=演算子（小さいか等しい）
  virtual bool operator>=(const S0_OrderedSet&) = 0;// >=演算子（大きいか等しい）
};
```

3 加法半群型

3.1 加法半群型 (abelian semigroup)

加法半群型は、加法半群、すなわち集合上に加法が定義されていてそれが結合法則を満たすものを表現します。加法という言葉は乗法と区別するために付けたもので、一般的な半群よりさらに強く可換性を仮定しています。演算がある法則を満たすことを C++ ではうまく表現することはできませんので、演算が定義されていることだけを表現することにします。

```
class S0_AbelianSemiGroup : public S0_Set
{
public:
  virtual S0_AbelianSemiGroup& operator+=(const S0_AbelianSemiGroup&) = 0;
// +=演算子（加法代入）
/*
  virtual S0_AbelianSemiGroup operator+(const S0_AbelianSemiGroup&) = 0;
// + 演算子（加法）
*/
};
```

上で + 演算子を定義する純粋仮想関数をコメントにしてありますが、これは残念ながら抽象クラス内で実体を生成するということでエラーになるようです。以下、エラーとなる純粋仮想関数はコメントにしてあります。

3.2 加法単位的半群型 (abelian monoid)

加法単位的半群型は、加法単位的半群、すなわち単位元 0 をもつ加法単位的半群を表現します。

```
class S0_AbelianMonoid : public S0_AbelianSemiGroup
{
public:
/*
  virtual const S0_AbelianMonoid Zero() const = 0; // 単位元 0
*/
};
```

3.3 加法群型 (abelian group)

加法群型は、加法群、すなわち任意の元がその逆元をもつ加法単位的半群を表現します。

```
class S0_AbelianGroup : public S0_AbelianMonoid
{
public:
/*
  virtual const S0_AbelianGroup operator-() const = 0; // 逆元
  virtual const S0_AbelianGroup operator-(const S0_AbelianGroup&) = 0;
// -演算子（減法）
*/
  virtual S0_AbelianGroup& operator-=(const S0_AbelianGroup&) = 0;
// -=演算子（減法代入）
};
```

4 半群型

4.1 半群型 (semigroup)

半群型は、半群、すなわち集合上に乗法が定義されていてそれが結合法則を満たすものを表現します。

```
class S0_SemiGroup : public S0_Set
{
public:
  virtual S0_SemiGroup& operator*=(const S0_SemiGroup&) = 0;
// *=演算子（乗法代入）
/*
  virtual const S0_SemiGroup operator*(const S0_SemiGroup&) = 0;
// *演算子（乗法）
*/
};
```

4.2 単位的半群型 (monoid)

単位的半群型は、単位的半群、すなわち単位元 1 をもつ単位的半群を表現します。

```
class S0_Monoid : public S0_SemiGroup
{
public:
/*
  virtual const S0_Monoid One() const = 0; // 単位元 1
*/
};
```

4.3 群型 (group)

群型は、群、すなわち任意の元がその逆元をもつ単位的半群を表現します。

```
class S0_Group : public S0_Monoid
{
public:
/*
  virtual const S0_Group Inv() const = 0; // 逆元
  virtual const S0_Group operator/(const S0_Group&) = 0;
// /演算子（除法）
*/
  virtual S0_Group& operator/=(const S0_Group&) = 0;
// /=演算子（除法代入）
};
```

5 環型

5.1 環型 (ring)

環型は、環、すなわち加法群上に乗法が定義されていて、それに関して単位的半群であり、乗法が加法に対して両側から分配的であるものを表現します。分配法則は C++ ではうまく表現できませんので、加法群型と単位的半群型を継承することのみが環型の表現となります。

```
class S0_Ring : public S0_AbelianGroup, public S0_Monoid
{
};
```

5.2 順序環型 (ordered ring)

順序環型は、順序環を表現します。順序環とは可換環上に順序付け (ordering) が与えられているものです。順序付けにより正と負の区別がつけられ、これにより自然に絶対値が導入されます。C++ ではこの絶対値が定義されていることを表現します。順序環の標数は 0 であり、自然に整数を埋め込むことができます。

```
class SO_OrderedRing : public SO_Ring, public SO_OrderedSet
{
public:
/*
  virtual const SO_OrderedRing Abs() const = 0; // 絶対値
*/
};
```

6 体型

6.1 体型 (field)

体型は、体、すなわち可換環において単位元 0 以外の元がすべて逆元をもつものを表現します。群型と違って乗法に関する逆元をとる関数と除法関数については 0 が引数として指定されたときのエラー処理を付け加えなければなりません。

```
class SO_Field : public SO_AbelianGroup, public SO_Group
{
/*
  virtual const SO_Field Inv() const = 0; // 逆元
  virtual const SO_Field operator/(const SO_Field&) = 0;
// /演算子（除法）
*/
  virtual SO_Field& operator/=(const SO_Field&) = 0;
// /=演算子（除法代入）
};
```

6.2 順序体型 (ordered field)

順序体型は、順序体を表現します。順序体の標数は 0 であり、自然に有理数を埋め込むことができます。

```
class SO_OrderedField : public SO_Field, public SO_OrderedSet
{
public:
/*
  virtual const SO_OrderedField Abs() const = 0; // 絶対値
*/
};
```

7 代数的構造の表現方法

7.1 数

整数型 (integer)

整数型は順序環型を基底クラスとします。

有理数型 (rational number)
　有理数型は順序体型を基底クラスとします。

実数型 (real number)
　実数型は順序体型を基底クラスとします。

7.2　加群、ベクトル空間、多元環

加群型 (module)
　加群型は加法群型を基底クラスとし、作用する環は環型を基底クラスとします。スカラー倍はフレンド関数として定義します。

ベクトル空間型 (vector space)
　ベクトル空間型は加法群型を基底クラスとし、作用する体は体型を基底クラスとします。スカラー倍はフレンド関数として定義します。

多元環型 (algebra)
　多元環型は環型を基底クラスとし、作用する体は体型を基底クラスとします。スカラー倍はフレンド関数として定義します。

複素数型 (complex number)
　複素数型は実数型上の多元環型として表現します。

四元数体型 (quaternion field)
　四元数体型は実数型上の多元環型として表現します。

参考文献

代数系

[1] 松坂 和夫, “代数系入門”, 岩波書店, 1976
[2] 日本数学会, “岩波数学辞典第 3 版”, 岩波書店, 1991